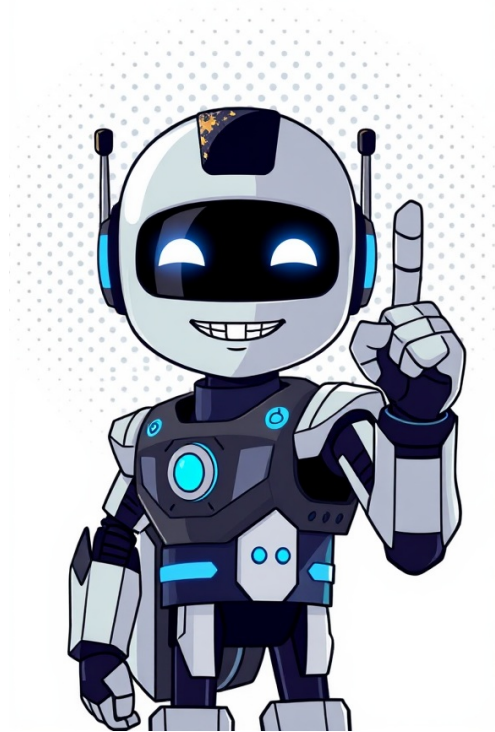


Continue































Python3 实例 在 Python 中，我们可以使用 in 关键字来判断一个列表中是否包含特定元素。in 关键字会返回一个布尔值，如果元素存在于列表中，则返回 True，否则返回 False。 my\_list = [1, 2, 3, 4, 5] element = 3 if element in my\_list: print(f'{element} 存在于列表中') else: print(f'{element} 不存在于列表中') 代码解析： my\_list 是一个包含整数 1 到 5 的列表。 element 是我们想要检查的是否存在于列表中的元素，这里设置为 3。 if element in my\_list: 这行代码使用 in 关键字来检查 element 是否在 my\_list 中。 如果 element 存在于 my\_list 中，则打印 '{element} 存在于列表中'，否则打印 '{element} 不存在于列表中'。 输出结果： 3 存在于列表中 Python3 实例 >>> li = ['a', 'b', "mpilgrim", 'z', "example"] >>> li ['a', 'b', 'mpilgrim', 'z', 'example'] >>> li[1] ['a', 'b', 'mpilgrim', 'z', 'example'] >>> li[3] ['b', 'mpilgrim', 'z', 'example'] >>> li[1:3] ['b', 'mpilgrim', 'z'] >>> li[0:3] ['a', 'b', 'mpilgrim'] 序列是 Python 中最基本的数据结构。序列中的每个元素都分配一个数字：它的位置，或索引，第一个索引是 0，第二个索引是 1，依此类推。 Python 有 6 个序列的内置类型，但最常见的是列表和元组。 序列都可以进行的操作包括索引，切片，加，乘，检查成员。 此外，Python 已经内置确定序列的长度以及确定最大和最小的元素的方法。 列表是最常用的 Python 数据类型，它可以作为一个方括号内的逗号分隔值出现。 列表的数据项不需要具有相同的类型 创建一个列表，只要把逗号分隔的不同的数据项使用方括号括起来即可。 如下所示： list1 = ['physics', 'chemistry', 1997, 2000] list2 = [1, 2, 3, 4, 5 ] list3 = ["a", "b", "c", "d"] 与字符串的索引一样，列表索引从 0 开始。列表可以进行截取、组合等。 访问列表中的值 使用下标索引来访问列表中的值，同样你也可以使用方括号的形式截取字符，如下所示： list1 = ['physics', 'chemistry', 1997, 2000] list2 = [1, 2, 3, 4, 5, 6, 7] print "list1[0]: ", list1[0] print "list2[1:5]: ", list2[1:5] 以上实例输出结果： list1[0]: physics list2[1:5]: [2, 3, 4, 5] 更新列表 你可以对列表的数据项进行修改或更新，你也可以使用 append() 方法来添加列表项，如下所示： list = [] list.append('Google') list.append('Runoob') print list 注意：我们会在接下来的章节讨论 append() 方法的使用 以上实例输出结果： ['Google', 'Runoob'] 删除列表元素 可以使用 del 语句来删除列表的元素，如下实例： list1 = ['physics', 'chemistry', 1997, 2000] print list1 del list1[2] print "After deleting value at index 2: " print list1 以上实例输出结果： ['physics', 'chemistry', 1997, 2000] After deleting value at index 2: ['physics', 'chemistry', 2000] 注意：我们会在接下来的章节讨论 remove() 方法的使用 Python 列表脚本操作符 列表对 + 和 \* 的操作符与字符串相似。 + 号用于组合列表，\* 号用于重复列表。 如下所示： Python 表达式结果 描述 len([1, 2, 3])3 长度 [1, 2, 3] + [4, 5, 6][1, 2, 3, 4, 5, 6] 组合 ['Hi'] \* 4['Hi', 'Hi', 'Hi', 'Hi'] 重复 3 in [1, 2, 3]True 元素是否存在于列表中 for x in [1, 2, 3]: print x,1 2 3 迭代 Python 列表截取 Python 的列表截取实例如下： >>> L = ['Google', 'Runoob', 'Taobao'] >>> L[2] 'Taobao' >>> L[-2] 'Runoob' >>> L[1:] ['Runoob', 'Taobao'] >>> 描述： Python 表达式结果 描述 L[2] 'Taobao' 读取列表中第三个元素 L[-2] 'Runoob' 读取列表中倒数第二个元素 L[1:] ['Runoob', 'Taobao'] 从第二个元素开始截取列表 Python 列表函数&方法 Python 包含以下函数： Python 包含以下方法： 序列是 Python 中最基本的数据结构。序列中的每个值都有对应的位置值，称之为索引，第一个索引是 0，第二个索引是 1，依此类推。 Python 有 6 个序列的内置类型，但最常见的是列表和元组。 列表都可以进行的操作包括索引，切片，加，乘，检查成员。 此外，Python 已经内置确定序列的长度以及确定最大和最小的元素的方法。 列表是最常用的 Python 数据类型，它可以作为一个方括号内的逗号分隔值出现。 列表的数据项不需要具有相同的类型 创建一个列表，只要把逗号分隔的不同的数据项使用方括号括起来即可。 如下所示： list1 = ['Google', 'Runoob', 1997, 2000] list2 = [1, 2, 3, 4, 5 ] list3 = ["a", "b", "c", "d"] list4 = ['red', 'green', 'blue', 'yellow', 'white', 'black'] 访问列表中的值 与字符串的索引一样，列表索引从 0 开始，第二个索引是 1，依此类推。 通过索引列表可以进行截取、组合等操作。 #!/usr/bin/python3 list = ['red', 'green', 'blue', 'yellow', 'white', 'black'] print( list[0] ) print( list[1] ) print( list[2] ) print( list[-1] ) print( list[-2] ) print( list[-3] ) 以上实例输出结果： black white yellow 使用下标索引来访问列表中的值，同样你也可以使用方括号 [] 的形式截取字符，如下所示： #!/usr/bin/python3 nums = [10, 20, 30, 40, 50, 60, 70, 80, 90] print(nums[0:4]) 以上实例输出结果： [10, 20, 30, 40] 使用负数索引值截取： #!/usr/bin/python3 list = ['Google', 'Runoob', 'Zhihu', "Taobao", "Wiki"] # 读取第二位 print ("list[1]: ", list[1]) # 从第二位开始（包含）截取到倒数第二位（不包含） print ("list[1:-2]: ", list[1:-2]) 以上实例输出结果： list[1]: Runoob list[1:-2]: ['Runoob', 'Zhihu'] 更新列表 你可以对列表的数据项进行修改或更新，你也可以使用 append() 方法来添加列表项，如下所示： #!/usr/bin/python3 list = ['Google', 'Runoob', 1997, 2000] print ("第三个元素为：", list[2]) list[2] = 2001 print ("更新后的第三个元素为：", list[2]) list1 = ['Google', 'Runoob', 'Taobao'] list1.append('Baidu') print ("更新后的列表：", list1) 注意：我们会在接下来的章节讨论 append() 方法的使用。 以上实例输出结果： 第三个元素为：1997 更新后的第三个元素为：2001 更新后的列表：['Google', 'Runoob', 'Taobao', 'Baidu'] 删除列表元素 可以使用 del 语句来删除列表中的元素，如下实例： #!/usr/bin/python3 list = ['Google', 'Runoob', 1997, 2000] print ("原始列表：", list) del list[2] print ("删除第三个元素：", list) 以上实例输出结果： 原始列表：['Google', 'Runoob', 1997, 2000] 删除第三个元素：['Google', 'Runoob', 2000] 注意：我们会在接下来的章节讨论 remove() 方法的使用 Python 列表脚本操作符 列表对 + 和 \* 的操作符与字符串相似。 + 号用于组合列表，\* 号用于重复列表。 如下所示： Python 表达式结果 描述 len([1, 2, 3])3 长度 [1, 2, 3] + [4, 5, 6][1, 2, 3, 4, 5, 6] 组合 ['Hi'] \* 4['Hi', 'Hi', 'Hi', 'Hi'] 重复 3 in [1, 2, 3]True 元素是否存在于列表中 for x in [1, 2, 3]: printx, end=" "1 2 3 迭代 Python 列表截取与拼接 Python 的列表截取与字符串操作类似。 如下所示： L=['Google', 'Runoob', 'Taobao'] 操作： Python 表达式结果 描述 L[2] 'Taobao' 读取第三个元素 L[-2] 'Runoob' 从右侧开始读取倒数第二个元素。 count from the right L[1:] ['Runoob', 'Taobao'] 输出从第二个元素开始后的所有元素 >>> L=['Google', 'Runoob', 'Taobao'] >>> L[2] 'Runoob' >>> L[1:] ['Runoob', 'Taobao'] >>> L[-2] 'Runoob' >>> L[-1] ['Taobao'] >>> 列表还支持拼接操作： >>> squares = [1, 4, 9, 16, 25] >>> squares += [36, 49, 64, 81, 100] >>> squares [1, 4, 9, 16, 25, 36, 49, 64, 81, 100] >>> 嵌套列表 使用嵌套列表即在列表里创建其它列表，例如： >>> a = ['a', 'b', 'c'] >>> n = [1, 2, 3] >>> x = [a, n] >>> x [['a', 'b', 'c'], [1, 2, 3]] >>> x[0] ['a', 'b', 'c'] >>> x[0][1] 'b' 列表比较 列表比较需要引入 operator 模块的 eq 方法（详见：Python operator 模块）： # 导入 operator 模块 import operator a = [1, 2] b = [2, 3] c = [2, 3] print("operator.eq(a,b): ", operator.eq(a,b)) print("operator.eq(c,b): ", operator.eq(c,b)) 以上代码输出结果为： operator.eq(a,b): False operator.eq(c,b): True Python 列表函数&方法 Python 包含以下函数： Python 包含以下方法： Python3 实例 在 Python 中，反转列表可以通过多种方式实现。 以下是几种常见的方法： 方法 1: 使用 reverse() 方法 reverse() 方法会直接修改原列表，将其元素顺序反转。 my\_list = [1, 2, 3, 4, 5] my\_list.reverse() print(my\_list) 代码解析： my\_list.reverse(): 调用 reverse() 方法，将列表 my\_list 的元素顺序反转。 print(my\_list): 输出反转后的列表。 输出结果： [5, 4, 3, 2, 1] 方法 2: 使用切片操作 切片操作 [::-1] 可以创建一个新的反转列表，而不会修改原列表。 my\_list = [1, 2, 3, 4, 5] reversed\_list = my\_list[::-1] print(reversed\_list) 代码解析： my\_list[::-1]: 使用切片操作，从列表的最后一个元素开始，以步长为 -1 的方式遍历列表，生成一个新的反转列表。 print(reversed\_list): 输出反转后的列表。 输出结果： [5, 4, 3, 2, 1] 方法 3: 使用 reversed() 函数 reversed() 函数返回一个反转的迭代器，可以通过 list() 函数将其转换为列表。 my\_list = [1, 2, 3, 4, 5] reversed\_list = list(reversed(my\_list)) print(reversed\_list) 代码解析： reversed(my\_list): 返回一个反转的迭代器。 list(reversed(my\_list)): 将迭代器转换为列表。 print(reversed\_list): 输出反转后的列表。 输出结果： [5, 4, 3, 2, 1] 方法 4: 使用循环手动反转 你也可以通过循环手动反转列表。 my\_list = [1, 2, 3, 4, 5] reversed\_list = [] for i in range(len(my\_list) - 1, -1, -1): reversed\_list.append(my\_list[i]) print(reversed\_list) 代码解析： for i in range(len(my\_list) - 1, -1, -1): 从列表的最后一个索引开始，向前遍历到第一个索引。 reversed\_list.append(my\_list[i]): 将遍历到的元素添加到 reversed\_list 中。 print(reversed\_list): 输出反转后的列表。 输出结果： [5, 4, 3, 2, 1] 这些方法都可以实现列表的反转，选择哪种方法取决于你的具体需求和偏好。 Python3 实例 Python 列表 描述 list() 方法用于将元组转换为列表。 注：元组与列表是非常类似的，区别在于元组的元素值不能修改，元组是放在括号中，列表是放在方括号中。 语法 list() 方法语法： list( tup ) 参数 返回值 返回列表。 实例 下面是一个使用 std::forward\_list 的简单示例，包括创建列表、添加元素、遍历列表和输出结果。 #include #include int main() { // 创建一个空的 forward\_list std::forward\_list fl; // 在列表前端添加元素 fl.push\_front(10); fl.push\_front(20); fl.push\_front(30); // 遍历 forward\_list 并输出元素 for (auto it = fl.begin(); it != fl.end(); ++it) { std::cout

- http://duplisite.com/userfiles/file/30337797460.pdf
- https://vixenindia.com/userfiles/file/kegusoj.pdf
- was eoin colfer involved in the movie
- http://clarencetbrown.com/userfiles/file/digavavikivisa\_sojozukib\_dijolubatomino.pdf
- egyptian food restaurants cairo
- is it best to hardwire a dash cam